FORM PTO-1390 (REV 12-29-99) | U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE

TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. 371

ATTORNEY'S DOCKET NUMBER: T2147-906522

U.S. APPLICATION NO. (If known, see 37 CFR 1.5): 09/582701

INTERNATIONAL APPLICATION NO.	INTERNATIONAL FILING DATE	PRIORITY DATE CLAIMED
PCT/FR99/02630	28/10/1999	30/10/1998

TITLE OF INVENTION: DERIVATION D'UNE CLASSE D'OBJETS PAR HERITAGE, INSTANCIATION OU CLONAGE

APPLICANT(S) FOR DO/EO/US: Armand NACHEF & Gerard SITBON

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. [x] This is a **FIRST** submission of items concerning a filing under 35 U.S.C. 371.
2. [ ] This is a **SECOND** or **SUBSEQUENT** submission of items concerning a filing under 35 U.S.C. 371.
3. [x] This express request to begin national examination procedures (35 U.S.C. 371(**f**) at any time rather than delay examination until the expiration of the applicable time limit set in 35 U.S.C.371(b) and PCT Articles 22 and 39(1).
4. [x] A proper Demand for International Preliminary Examination was made by the 19th month from the earliest claimed priority date.
5. [x] A copy of the International Application as filed (35 U.S.C. 371(c)(2))
   a. [ ] is transmitted herewith (required only if not transmitted by the International Bureau).
   b. [x] has been transmitted by the International Bureau.
   c. [ ] is not required, as the application was filed in the United States Receiving Ofice (RO/US).
6. [ ] A translation of the International Application into English (35 U.S.C. 371(c)(2)).
7. [ ] Amendments to the claims of the International Application under PCT Article 19(35 U.S.C. 371(c)(3))
   a. [ ] are transmitted herewith (required only if not transmitted by the International Bureau).
   b. [ ] have been transmitted by the International Bureau.
   c. [ ] have not been made; however, the time limit for making such amendments has NOT expired.
   d. [ ] have not been made and will not be made.
8. [ ] A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).
9. [x] An oath or declaration of the inventor(s) (35 U.S.C. 371(c)(4)).
10. [ ] A translation of the annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371(c)(5)).

Items 11. to 16. below concern document(s) or information included:

11. [x] An Information Disclosure Statement under 37 CFR 1.97 and 1.98. with copies of cited references
12. [x] An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included. to BULL S.A.
13. [x] A FIRST preliminary amendment.
    [ ] A SECOND or SUBSEQUENT preliminary amendment.
14. [ ] A substitute specification.
15. [x] A change of power of attorney and/or address letter.
16. [x] Other items or information:

PCT FORMS: PCT/IB/301 & 308; PCT/RO/101

532 Rec'd PCT/PTC 30 JUN 2000

U.S. APPLICATION NO (if known, see 37 CFR 1.5) 09/582701 | INTERNATIONAL APPLICATION NO PCT/FR99/02630 | ATTORNEY'S DOCKET NUMBER T2147-906522

17. [x] The following fees are submitted:

	CALCULATIONS PTO USE ONLY
BASIC NATIONAL FEE (37 CFR 1.492 (a) (1) - (5)):	
Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1.445(a)(2)) paid to USPTO and International Search Report not prepared by the EPO or JPO **$970.00**	
International preliminary examination fee (37 CFR 1.482) not paid to USPTO but International Search Report prepared by the EPO or JPO **$840.00**	
International preliminary examination fee (37 CFR 1.482) not paid to USPTO but international search fee (37 CFR 1.445(a)(2)) paid to USPTO **$690.00**	
International preliminary examination fee paid to USPTO (37 CFR 1.482) but all claims did not satisfy provisions of PCT Article 33(1)-(4) **$670.00**	
International preliminary examination fee paid to USPTO (37 CFR 1.482) and all claims satisfied provisions of PCT Article 33(1)-(4) **$96.00**	
ENTER APPROPRIATE BASIC FEE AMOUNT =	$ 840.00
Surcharge of **$130.00** for furnishing the oath or declaration later than [ ] 20 [ ] 30 months from the earliest claimed priority date (37 CFR 1.492(e)).	$

CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE	
Total claims	- 20 =		X **$18.00**	$
Independent claims	- 3 =		X **$78.00**	$
MULTIPLE DEPENDENT CLAIM(S) (if applicable)			+ **$260.00**	$
TOTAL OF ABOVE CALCULATIONS =				$ 840.00
Reduction of 1/2 for filing by small entity, if applicable. A Small Entity Statement must also by filed (Note 37 CFR 1.9, 1.27, 1.28).				$
SUBTOTAL =				$ 840.00
Processing fee of **$130.00** for furnishing the English translation later than [ ] 20 [ ] 30 months from the earliest claimed priority date (37 CFR 1.492(f)).			+	$ 130.00
TOTAL NATIONAL FEE =				$ 970.00
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31). **$40.00** per property			+	$ 40.00
TOTAL FEES ENCLOSED =				$1010.00
				Amount to be refunded: $
				charged: $

a. [x] A check in the amount of $1010.00 to cover the above fees is enclosed.

b. [ ] Please charge my Deposit Account No. _____ in the amount of $_____ to cover the above fees. A duplicate copy of this sheet is enclosed.

c. [X] The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Deposit Account No. 501165. A duplicate copy of this sheet is enclosed.

NOTE: Where an appropriate time limit under 37 CFR 1.494 or 1.495 has not been met, a petition to revive (37 CFR 1.137(a) or (b)) must be filed and granted to restore the application to pending status.

SEND ALL CORRESPONDENCE TO:
Edward J. Kondracki
MILES & STOCKBRIDGE P.C.
1751 Pinnacle Drive, Suite 500
McLean, VA 22102-3833

SIGNATURE: Edward J. Kondracki

NAME: Edward J. Kondracki

REGISTRATION NUMBER: 20,601

June 30, 2000

T2147-906522-US3681/HD(PCT)-Nachef

IN THE UNITED STATES DESIGNATED/ELECTED OFFICE (D.O./E.O./US)

Applicants:	Armand NACHEF et al.
International Application No.:	PCT/FR99/02630
International Filing Date:	28/10/1999
U.S. Serial No.:	09/582,701
U.S. Filing Date:	June 30, 2000
For:	DERIVING AN OBJECT CLASS BY INHERITANCE, INSTANCIATION OR CLONING

McLean, Virginia

SUPPLEMENTAL PRELIMINARY AMENDMENT

Honorable Commissioner of Patents and Trademarks
Washington, DC 20231

Sir:.

Please amend the subject application, filed concurrently herewith, as indicated below:

IN THE TITLE:

Please correct the spelling from "INSTANCIATION" to –INSTANTIATION--.

IN THE SPECIFICATION:

Page 1, at line 2 following the title, insert the following heading at the left hand margin followed by the following paragraph:

--Cross reference to Related Applications:--

The subject matter of this application is related to

(1) application Serial No. 09/582,755, filed on June 30, 2000, in the names of Armand NACHEF, Gerard STIBON and Jean-Michel RAVON, entitled "METHOD FOR CONTROLLING A FUNCTION EXECUTABLE BY SPECIFIC COMMANDS TO DIFFERENT SOFTWARE TOOLS", and corresponding to French Application No. 98 13645 and PCT application No. PCT/FR99/02629,

(2) application Serial No.09/582,762, filed on June 30, 2000, in the name of Jean-Marc GOUBE, Armand NACHEF and Gerard SITBON, entitled " METHOD FOR GENERATING INTERFACES FOR CONTROL BY A COMPUTER SYSTEM USER", and corresponding to French Application No. 98/13642 and PCT application No. PCT/FR99/02632;

(3) application Serial No. 09/582,702, filed on June 30, 2000, in the names of Armand NACHEF and Jean-Michel RAVON, entitled "METHOD FOR AUTOMATICALLY GENERATING IN AN OBJECT-ORIENTED LANGUAGE A DRIVER FOR IMPLEMENTING AN ABSTRACT METHOD OF AN ABSTRACT CLASS and corresponding to French Application No. 98/13644and PCT application No. PCT/FR99/02633, and

(4) application Serial No. 09/582,757, filed on June 30, 2000, in the names of Armand NACHEF and Gerard SITBON, entitled "DYNAMIC CREATION OF OBJECT CLASSES" and corresponding to French Application No. 98/13641 and PCT

application No. PCT/FR99/02634,

the subject matter of each of said applications is hereby incorporated by reference.--

Page 1, at line 4, and before the first paragraph, delete "Technical Field" insert the following heading at the left-hand margin:

--FIELD OF THE INVENTION--;

Page 1, at line 11, and before the second paragraph, delete "The Prior Art" and insert the following heading at the left-hand margin:

--DESCRIPTION OF RELATED ART--;

Page 2, at line 18 and before the paragraph beginning "Fig. 1...", delete "Presentation of the Drawings" and insert the following heading at the left hand margin:

--BRIEF DESCRIPTION OF THE DRAWINGS--;

Page 3, at line 12 and before the paragraph beginning "Fig. 1,...", delete "Detailed Description of Examples Illustrating the Invention" and insert the following heading at the left hand margin:

--DESCRIPTION OF THE PREFERRED EMBODIMENT(S)--;

Page 18, after line 25, insert the following new paragraph:

--While this invention has been described in conjunction with specific embodiments thereof, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the preferred embodiments of the invention as set forth herein, are intended to be illustrative, not limiting. Various

changes may be made without departing from the true spirit and full scope of the invention as set forth herein and defined in the claims.—

IN THE CLAIMS:

Please cancel claims 1 - 10 in their entirety and without prejudice and substitute the following new claims:

--11. A method for deriving a class and/or an object having a first given name (class 1), comprising making a copy (27c) of an entire tree (27a) of the class or the object, storing (D) the copy of the tree and changing said first given name in order to assign a second name (class D2) to the stored copy.

12. A method according to claim 11, wherein the copy is made through a serialization of the tree representing said class or said object by copying the tree into a memory (D), and storing the copy of the tree consists of copying it again into memory (30).

13. A method according to claim 11, wherein the derivation is an inheritance of the class (class1).

14. A method according to claim 11, wherein the derivation is an instantiation of the class (class1).

15. A method according to claim 11, wherein the derivation is a cloning of an object.

16. A method according to claim 11, further comprising automatically generating the class or the derived object by means of a tool (30) having at least one dialog box (21).

17. A method according to claim 16, further comprising implementing the derivation by a computer designer (C), and using a command interface (11) of a computer system (10) used for control of the computer system by a user (U).

18. A method according to claim 12, wherein the derivation is an inheritance of the class (class1).

19. A method according to claim 12, wherein the derivation is an instantiation of the class (class1).

20. A method according to claim 12, wherein the derivation is a cloning of an object.

21. A method according to claim 12, further comprising automatically generating the class or the derived object by means of a tool (30) having at least one dialog box (21).

22. A method according to claim 13, further comprising automatically generating the class or the derived object by means of a tool (30) having at least one dialog box (21).

23. A method according to claim 14, further comprising automatically generating the class or the derived object by means of a tool (30) having at least one dialog box (21).

24. A method according to claim 15, further comprising automatically generating the class or the derived object by means of a tool (30) having at least one dialog box (21).

25. A method according to claim 21, further comprising implementing the derivation by a computer designer (C), and using a command interface (11) of a computer system (10) used for control of the computer system by a user (U).

26. A method according to claim 22, further comprising implementing the derivation by a computer designer (C), and using a command interface (11) of a computer system (10) used for control of the computer system by a user (U).

27. A method according to claim 23, further comprising implementing the derivation by a computer designer (C), and using a command interface (11) of a computer system (10) used for control of the computer system by a user (U).

28. A method according to claim 24, further comprising implementing the derivation by a computer designer (C), and using a command interface (11) of a computer system (10) used for control of the computer system by a user (U).

29. A computer system for implementing a method for deriving a class and/or an object having a first given name (class 1), comprising making a copy (27c) of an entire tree (27a) of the class or the object, storing (D) the copy of the tree and changing said first given name in order to assign a second name (class D2) to the stored copy.

30. A computer system according to claim 29, wherein the copy is made through a serialization of the tree representing said class or said object by copying the

tree into a memory (D), and storing the copy of the tree consists of copying it again into memory (30).

31. A computer system according to claim 29, wherein the derivation is an inheritance of the class (class1).

32. A computer system according to claim 29, wherein the derivation is an instantiation of the class (class1).

33. A computer system according to claim 29, wherein the derivation is a cloning of an object.

34. A computer system according to claim 29, further comprising automatically generating the class or the derived object by means of a tool (30) having at least one dialog box (21).

35. A method according to claim 29, further comprising implementing the derivation by a computer designer (C), and using a command interface (11) of a computer system (10) used for control of the computer system by a user (U).

36. A computer system according to claim 29, further including a command interface (11), for implementing the method.

37. A computer system according to claim 29, wherein the command interface includes a design module (13) for implementing the method by a designer (C) and further including a console (17) for the control of the computer system by a user (U).--

IN THE ABSTRACT:

Please cancel the Abstract at page 21 and substitute the following new Abstract:

--ABSTRACT

An object class and/or an object having a given name (class1) is derived by making a copy, preferably through serialization, of the entire tree (27a) of the class or the object, by storing the copy of the tree on a disk D and by assigning a name (classD2) to the stored copy.--

REMARKS

This Supplemental Preliminary Amendment is filed to insert headings to conform the application to U.S. practice, to correct informalities in the specification, claims and abstract resulting from a literal translation of the French text, and to eliminate the use of multiple dependent claims.

Early action on the merits is earnestly solicited.

Respectfully submitted,

MILES & STOCKBRIDGE P.C.

Date: August 28, 2000

By: Edward J. Kondracki
Edward J. Kondracki
Registration No. 20,604

1751 Pinnacle Drive – Suite 500
McLean, VA 22102-3833
Tel.: 703/903-9000
Fax: 703/610-8686

T2147-906522-US3681/HD(PCT)-NACHEF

UNITED STATES DESIGNATED/ELECTED OFFICE (D.O./E.O./US)

Applicants:	Armand NACHEF et al.
International Application No.:	PCT/FR99/02630
International Filing Date:	28/10/1999
U.S. Serial No.:	To be Assigned
U.S. Filing Date:	June 30, 2000
For:	DERIVATION D'UNE CLASSE D'OBJETS PAR HERITAGE, INSTANCIATION OU CLONAGE

McLean, Virginia

PRELIMINARY AMENDMENT

Honorable Commissioner of Patents and Trademarks
Washington, DC 20231

Sir:.

This Preliminary Amendment is filed contemporaneously with the filing of the subject application. Please amend the claims of the application as indicated below without prejudice in order to be able to reintroduce the subject matter in translated claims.

IN THE CLAIMS:

Claim 3, line 1, delete "la revendication 1 ou 2" and replace with –revendication 1–.

Claim 4, line 1, delete "la revendication 1 ou 2" and replace with --revendication 1--.

Claim 5, line 1, delete "la revendication 1 ou 2", and replace with --revendication 1--.

Claim 6, line 1, delete "l'une des revendications 1 à 5" and replace with --revendication 1--.

Claim 8, line 1, delete "l'une des revendications 1 à 7" and replace with --revendication 1--.

REMARKS

This Amendment is made, without prejudice, to avoid and remove improper multiple dependency of the claims and the extra expense associated therewith. Upon translation of the application, the dependent claims will be reintroduced as singly dependent claims.

Respectfully submitted,

MILES & STOCKBRIDGE P.C.

Date: June 30, 2000

By: 
Edward J. Kondracki
Registration No. 20,604

1751 Pinnacle Drive – Suite 500
McLean, VA 22102-3833
Tel.: 703/903-9000
Fax: 703/610-8686

T2147-906522-US3681/HD(PCT)-NACHEF

IN THE UNITED STATES DESIGNATED/ELECTED OFFICE (D.O./E.O./US)

Applicants:	Armand NACHEF et al.
International Application No.:	PCT/FR99/02630
International Filing Date:	28/10/1999
U.S. Serial No.:	09/582,701
U.S. Filing Date:	June 30, 2000
For:	DERIVING AN OBJECT CLASS BY INHERITANCE, INSTANCIATION OR CLONING

PROPOSED DRAWING CORRECTIONS

Hon. Commissioner of Patents and Trademarks
Washington, D.C. 20231

Sir:

Applicant requests approval of the drawing corrections on Figs. 1 - 13 as shown in red on the attached five (5) sheets.

The proposed corrections only comprise labeling the boxes in Fig. 1 to conform the drawing to the specification and claims and removing the headings "1/5" to "5/5" to conform the drawings to U.S. practice.

Respectfully submitted,

MILES & STOCKBRIDGE P.C.

Date: August 28, 2000

By: Edward J. Kondracki
Edward J. Kondracki
Registration No. 20,604

1751 Pinnacle Drive – Suite 500
McLean, VA 22102-3833
Tel.: 703/903-9000

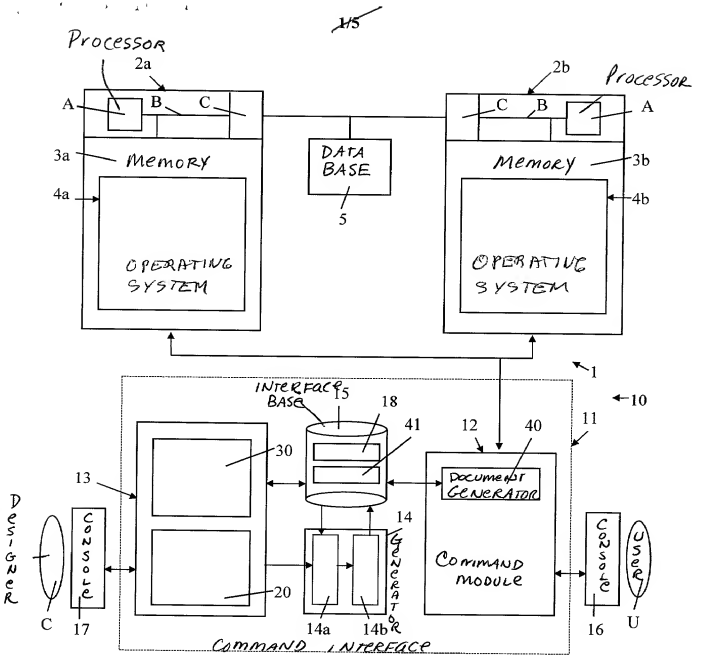
Processor
2a
A
B
C
3a
Memory
4a
OPERATING SYSTEM
DATA BASE
5
2b
Processor
C
B
A
Memory
3b
4b
OPERATING SYSTEM
1
10
INTERFACE BASE
15
18
41
12
40
11
30
13
DOCUMENT GENERATOR
14
GENERATOR
COMMAND MODULE
DESIGNER
CONSOLE
C
17
20
14a
14b
CONSOLE
USER
16
U
COMMAND INTERFACE

FIG. 1

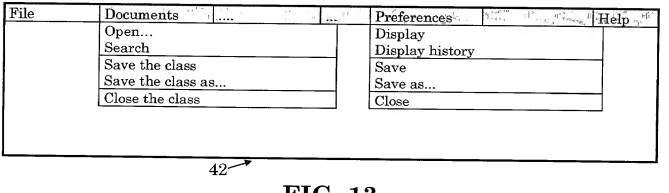
File
Documents
....
...
Preferences
Help
Open...
Search
Save the class
Save the class as...
Close the class
Display
Display history
Save
Save as...
Close
42

FIG. 13

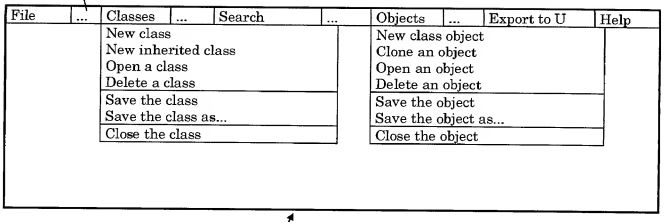
20
22
File
...
Classes
...
Search
...
Objects
...
Export to U
Help
New class
New inherited class
Open a class
Delete a class
Save the class
Save the class as...
Close the class
New class object
Clone an object
Open an object
Delete an object
Save the object
Save the object as...
Close the object
21

FIG. 2

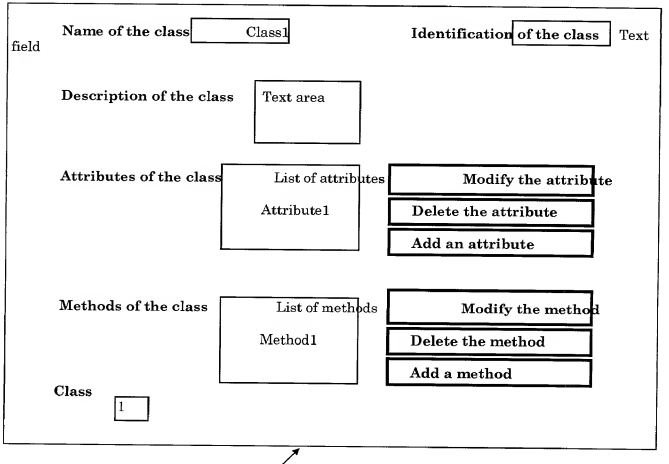
Name of the class
Class1
Identification of the class
Text field
Description of the class
Text area
Attributes of the class
List of attributes
Attribute1
Modify the attribute
Delete the attribute
Add an attribute
Methods of the class
List of methods
Method1
Modify the method
Delete the method
Add a method
Class
1
23
20

FIG. 3

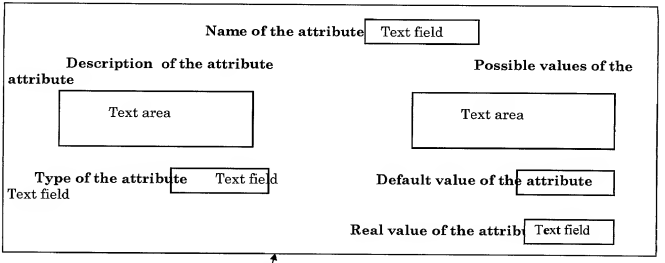

FIG.4

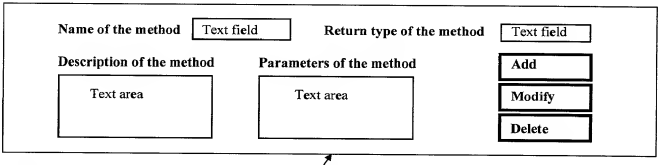

FIG. 5

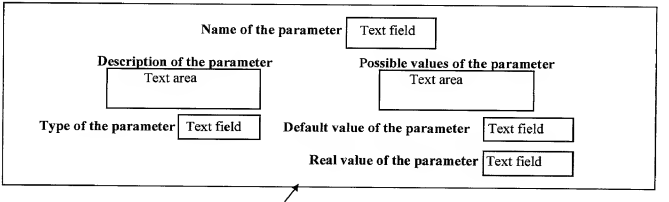

FIG.6

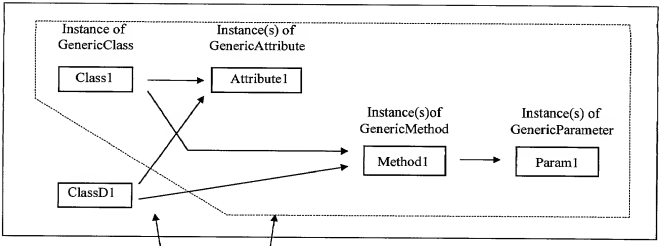

FIG. 7

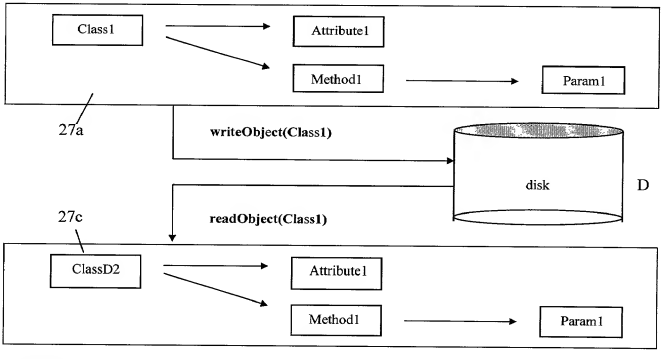

FIG. 8

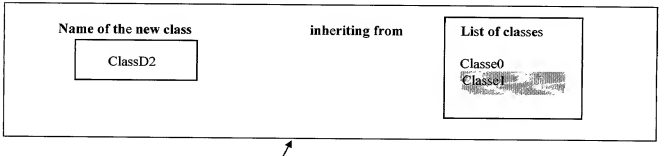

FIG. 9

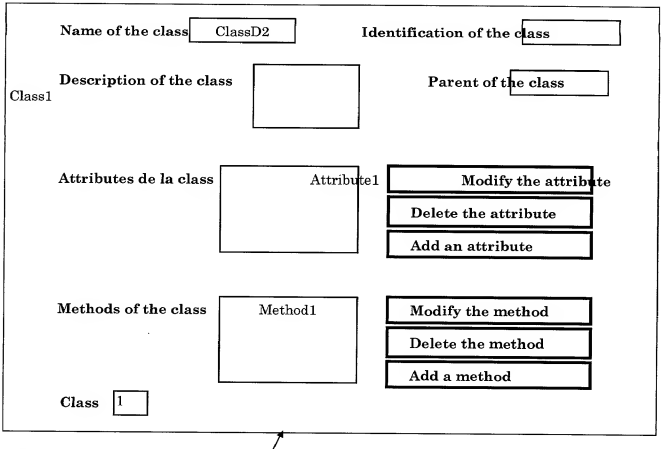

FIG. 10

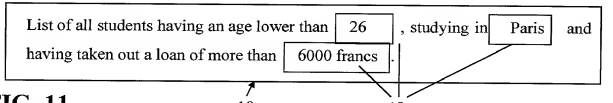

FIG. 11

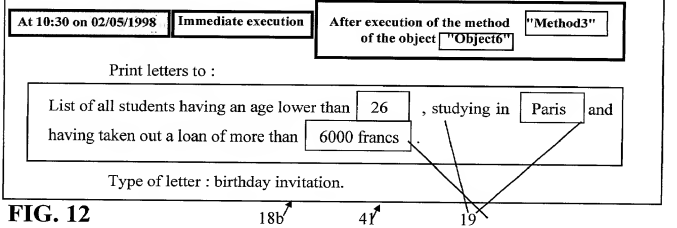

FIG. 12

DERIVING AN OBJECT CLASS BY INHERITANCE, INSTANCIATION OR CLONING

Technical Field

The invention relates to the derivation of an object class by inheritance, instantiation or cloning in a computer system. It applies to any computer program using an object-oriented language like those currently known as C++ and Java®, and to any computer system. Its subjects are a method for deriving an object class or an object by inheritance, instantiation or cloning, and the computer system that implements this method.

The Prior Art

An object class includes as members at least one attribute and at least one method. We will use the example of a class "print" related to the printing, in a computer system, of clients' bank account statements. The class can have as a first attribute a distribution list based on a set of criteria such as each client's current account balance, and as a second attribute, the number of copies of each client's statement. The class can also have a first method "print" having as a parameter a destination printer, and a second method "display" for displaying the current print jobs.

The duplication of a class is accomplished by pointing to the attributes and methods of the class. Consequently, the attributes and methods of the class are common to the original class and the duplicated class, in such a way that the two classes are independent of one another. In order to make them independent by creating a new class by inheritance, and/or to instantiate a class and/or to clone an object, it is necessary not only to duplicate the instance of the class, but also to duplicate all the instances directly or indirectly linked to the instance of the class. In the above example, it is first necessary to duplicate the class "print," the two attributes, the two methods and the parameter of the first method. Generally, it is necessary to create loops for duplicating the entire instance tree that forms the class or the object.

The drawbacks of the current derivation of an object class are now clear. It requires lengthy operations and a large amount of memory space. For example, the current duplication requires searching for the leaves, successively duplicating them, searching for the parents of each

of the leaves, duplicating them, linking the duplicated leaves to the respective parents that have been duplicated, and so on up to the root.

Summary of the Invention

One object of the invention is to rapidly perform a derivation of a class.

A second object is not to require a large amount of memory space.

The subject of the invention is a method for deriving a class and/or an object having a given name, characterized in that it consists of making a copy of the entire tree of the class or the object, of storing the copy of the tree and of changing said name in order to assign a name to the stored copy.

Preferably, the copy is made through a serialization of the tree representing said class or said object by copying the tree into a memory, and the storage of the copy of the tree consists of copying it again into memory.

The corollary subjects of the invention are the resulting computer system, and a recording medium such as a magnetic diskette or a CD-ROM, incorporating software for implementing the method.

Presentation of the Drawings

- Fig. 1 is a block diagram of a computer system that implements the method of the invention for deriving a class or an object.
- Fig. 2 is a window used by a designer to implement the method of the invention.
- Fig. 3 is a dialog box used by the designer to dynamically create a class that suits the use of the system desired by the user.
- Fig. 4 is a dialog box used by the designer to define an attribute of the class presented in Fig. 3.
- Fig. 5 is a dialog box used by the designer to define a method of the class presented in Fig. 3.
- Fig. 6 is a dialog box used by the designer to define a parameter of a method defined in Fig. 5.

♦ Fig. 7 is a block diagram of the tree structure of the class as defined by means of the dialog boxes presented in Figs. 3 through 6.

♦ Fig. 8 is a block diagram illustrating a method for the derivation of the class presented in Fig. 7.

♦ Figs. 9 and 10 are dialog boxes used by the designer to define the class derived by using the method illustrated in Fig. 8.

♦ Figs. 11 and 12 illustrate two respective interfaces resulting from the implementation of the method of the invention.

♦ Fig. 13 is a window presented to the user for the utilization of the interfaces illustrated in Figs. 1, 11 and 12.

Detailed Description of Examples Illustrating the Invention

Fig. 1 illustrates a computer system 10 operated by a user U by means of a command interface 11. The system 10 can be any type of system. The system 10 illustrated includes a computer infractructure 1 comprising at least one machine, two machines 2a and 2b in the example illustrated. Each machine illustrated has, in a way that is well known, at least one processor A connected through a bus B to a memory 3. Generally, the memory can be local or distributed, or the machines can form nodes of a network within the infrastructure 1. Software means, including one or more operating systems 4, are stored in each memory 3. In order to better highlight the advantages of the invention, the system will be considered to be a heterogeneous system, representing a case that is difficult to handle. The two machines 2 are assumed to run on two respective operating systems 4a and 4b of different types or versions, for example an operating system of the proprietary type and an operating system of the open type, for example one known by either of the registered trade names UNIX or Windows NT. The computer infrastructure 1 also has database means, called a database 5, which can be constituted by one or more local or remote databases. The machines 2 each also have an input/output interface C connected to the bus B. The input/output interfaces C of the two machines are connected to one another as well as to the database 5.

The command interface 11 comprises a command module 12, a design module 13, an interface generator 14 and an interface base 15. The module 12 is connected by a two-way

connection to the computer infrastructure 1. The interface base 15 is connected by a two-way connection to the modules 12 and 13 and to the generator 14. The generator 14 is also connected that it can be controlled by the module 13. The command module 12 is used by the user U to control and use the data of the system 10. The design module 13 is used by a designer C, who is another user of the interface 11 but who is a computer specialist. The two users U and C have respective consoles 16 and 17 attached to the respective modules 12 and 13.

In short, the user U defines needs in order to understand and improve the running of the company and submits his needs to the designer C. The designer C creates, by means of his module 13, software objects that meet the needs of the user U and sends them to the generator 14 to be converted into interfaces 18 (Figs. 11 and 12) that correspond to his needs and express the commands of the system in a language that is understandable to him. These interfaces are used by the user to create documents and are called document interfaces.

The user U is, for example, a bank branch manager who is not a computer specialist. It is assumed, for example, that the user U wants to consult the bank's computer infrastructure 1 to perform queries in order to obtain a list of the students that are clients of the branch, and print personalized information, invitation and reminder letters for them. The user U transmits his needs to the designer C, for example during a meeting between them. The designer transcribes these needs using the module 13, in a first step, to create, modify and/or delete objects and object classes related to these needs. For example, in order to meet the above-mentioned need of the user U, the designer C will create a class "print" with: a first attribute "distribution list" based on a set of criteria, in this case the students of the branch; a second attribute "number of copies" (integer); a first method "print" having as a parameter the destination printer; and a second method "display print jobs" in order to know the progress of the printing. The objects and classes created by the designer are stored in the interface base 15. The final transcription by the designer C of the needs of the user U is contained only in objects that have previously been directly created or that are instances of previously defined classes. These objects are contained in the base 15, which is contained in the command interface 11 in the example illustrated but which could be outside the interface 11 and included in the computer infrastructure 1. The objects are converted into document interfaces presented to the user U for him to use to create command documents that meet his needs.

The module 13 will be presented first. It uses two software tools 20 and 30, which respectively execute a process for dynamically creating classes and a process for deriving classes and objects. The tool 20 for dynamically creating classes will now be presented.

Fig. 2 illustrates an example of a screen 21 displayed on the console 17 that serves as an interface between the designer C and the module 13. The screen 21 has a menu bar 22, that includes in particular the menus "Classes," "Objects," "Search" and Export to U." The "Classes" menu makes it possible to dynamically create object classes, by inheritance or not. It also makes it possible to open a class, possibly in order to modify it, save it, or close it. The "Objects" menu makes it possible to instantiate the classes defined previously, in order to create objects, which can then be cloned. Just as for the classes, it is possible to open an object, modify it, save it on the disk and close it. With the tool 20, the designer C can define the commands that are found in both menus, as illustrated for example in those of the screen 21. The "Classes" menu illustrated contains the commands "New class," "New inherited class," "Open a class," "Delete a class," "Save the class," "Save the class as..." and "Close the class." The "Objects" menu contains the commands "New class object," "Clone an object," "Open an object," "Delete an object," "Save the object," "Save the object as..." and "Close the object."

The tool 20 implements a method for automatically generating an object class. The method is triggered automatically by the design module 13 when the designer C activates the command "New class" on the "Classes" menu in the window 22 of Fig. 2. The method for automatically generating a class consists of creating a global generic class "GenericClass" having two possible members, one of them being related to at least one attribute and the other being related to at least one method, at least one of the two members being an instance of a generic class, and of instantiating the global generic class in order to have said object class. In the tool 20 illustrated, two generic classes "GenericAttribute" and "GenericMethod" are assigned to the two respective members of the global generic class "GenericClass." Furthermore, if a method includes at least one parameter not fixed, this parameter is itself an instance of a corresponding generic class "GenericParameter."

Generally, the four generic classes thus created are not visible to the designer C. In the example illustrated, they are made available to the designer C when he executes a command in the "Classes" and "Objects" menus. Thus, a generic class is defined as a class that allows the

designer C to create a new class by creating an instance of a global generic class. Since the creation of instances is dynamic in all languages, the creation of a class is also dynamic for the designer C. Likewise, given that an object is a new instance of a class, the creation of an object also corresponds to the creation of a copy of this class, i.e., to a new instance of a global generic class. Consequently, the process triggered by the tool 20 is also triggered when the designer C activates the command "New class object" in the "Objects" menu in the window 22 of Fig. 2. The method will now be illustrated in the way that it is presented to the designer C by the tool 20.

Fig. 3 illustrates an example of a dialog box 23 that the tool 20 opens when the designer C clicks on the command "New class" in the "Classes" menu. The designer C uses the dialog box 23 to enter all of the data relative to a new class that does not inherit anything. It is understood that the data are the attributes and the methods of the new class.

The box 23 illustrated contains, for the entry of the data:

- a text field "Name of the class"
- a text field "Identification of the class (Class Id)"
- a text area "Description of the class"
- a list "Methods of the class" and
- a list "Attributes of the class".

The box 23 also contains the six command buttons outlined in bold lines:

- "Modify the method"
- "Delete the method"
- "Add a method"
- "Modify the attribute"
- "Delete the attribute" and
- "Add an attribute".

When this dialog box is validated, it results in the creation of an instance of a global generic class called "GenericClass." The global generic class of the example illustrated in Fig. 3 has several attributes.

The first attribute is a name, formed by a character string designating the name of the class. It is written by filling in the field "Name of the class" in the box 23.

The second attribute is an identification of the class, formed by a character string that makes it possible to uniquely identify the class or the object in the system. This attribute is written by filling in the field "Identification of the class" in the box 23, for example indicating the date and time of creation, a random number forming a ticket, etc.

The third attribute is a description, formed by the text that describes the purpose of the class. It is written by filling in the area "Description of the class."

The fourth attribute is a table named "Attributes," which references the attributes of the class. The attributes of the class are themselves instances of a generic attribute class called "Generic Attribute," which has as attributes:

- the name of the attribute
- the description of the attribute
- either the type of the attribute or the possible values of the attribute
- the default value of the attribute, given at the creation of the class "GenericAttribute" and
- the real value of the attribute, which is invisible to the designer C and is therefore not defined during the creation of the class. It is defined by the user U as described below.

For example, the class "Attributes[i]" is an instance of "Generic Attribute" and references the i^{th} attribute of the class defined by the designer C.

Fig. 4 illustrates a dialog box 24 of the tool 20 constituting the fourth attribute of "GenericClass." This box is opened from the box 23, for example by giving the name "Class1" to the class derived through the instantiation of the global generic class "GenericClass" and by activating the button "Add an attribute." The box 24 contains:

- a text field "Name of the attribute"
- a text area "Description of the attribute"
- a text field "Type of the attribute"
- a text area for "Possible values of the attribute"
- a text field "Default value of the attribute" and
- a text field "Real value of the attribute," shown in gray in Fig. 4 to indicate that it is not visible to the designer C.

Likewise, to modify an attribute, one need only select the attribute from the list of

attributes in Fig. 3 and activate the button "Modify the attribute," in order to bring up the dialog box 24 of Fig. 4 and modify the data that appears in the box 24.

As the first attribute, the attribute is given the name "Attribute1" and the other fields are filled in. The validation of the dialog box 24 results in the creation of an instance of the generic class "GenericAttribute." The name "Attribute1" will appear in the list of attributes in Fig. 3, as indicated.

The fifth attribute of the generic class "GenericClass" is a table named "Methods," which references the methods of the class. These methods, in the example illustrated, are also instances of a generic method class called "GenericMethod." A method is defined by the following attributes:

- its name
- its description
- its return type
- its parameter, and
- its execution script.

Fig. 5 illustrates a dialog box 25 of the tool 20 constituting the fifth attribute of "GenericClass." This box is opened from the box 23, for example by activating the button "Add a method." The box 25 contains:

- a text field "Name of the method"
- a text area "Description of the method"
- a text field "Return type of the method"
- a list of "Parameters of the method"
- a command button "Add"
- a command button "Modify," and
- a command button "Delete."

As the first method, the method is given a name, for example "Method1," and the other fields are filled in. The validation of the dialog box 25 results in the creation of an instance of the generic class "GenericMethod." The name "Method1" will appear in the list of methods in Fig. 3, as indicated.

The generic class "GenericMethod" contains, in addition to the attributes of the method, a

"Parameters" table that references the parameters of the method. The table may be empty if the generic class does not contain any parameter to be determined, or may not exist if this generic class is not intended to have any determinable parameter or if it has only fixed or predetermined parameters. In the table, the parameters are also instances of a generic class "GenericParameter." A parameter of a method is defined by the following attributes:

- its name
- its description
- either its type or its possible values
- its default value, and
- its real value, not visible to the designer C.

For example, if the parameter relates to the number of pages to be printed, the default value will be 1, but the user U could indicate another real value, for example 3 in order to have three copies of the pages to be printed.

Fig. 6 illustrates a dialog box 26 of the tool 20 for filling in the parameter table of "GenericMethod" in Fig. 5. The box 26 is opened from the box 25 by activating the button "Add." The box 26 contains:

- a text field "Name of the parameter"
- a text area "Description of the parameter"
- a text field "Type of the parameter"
- a text area "Possible values of the parameter"
- a text field "Default value of the parameter," and
- a text field "Real value of the parameter," represented in gray in Fig. 5 to indicate that this field is not visible to the designer C.

As a first parameter, it is given for example the name "Param1" and the other fields are filled in. The validation of the dialog box 26 results in the creation of an instance of the generic class "GenericParameter." The name "Param1" will appear in the list of parameters in Fig. 5.

The dialog box 23 of Fig. 3 being filled, the class "Class1" is created dynamically. It can be validated by being saved with the command "Save the class" or "Save the class as...".

Fig. 7 illustrates the tree structure 27a of the class "Class1" resulting from the validation of the box 23. In Fig. 7, the class "Class1" is an instance of the global generic class

"GenericClass" and has an attribute "Attribute1" and a method "Method1," itself having a parameter "Param1," all three of which are instances of three respective generic classes. Of course, another class that is an instance of the global generic class "GenericClass" could have several instances of each of the three generic classes "GenericAttribute," "GenericMethod" and "GenericParameter." This tree makes it possible to dynamically modify, add and delete members of the class (attributes or methods) at any time.

It is known that an object is an instance of a class. The activation of the command "New class object" in the "Objects" menu in the window 21 of Fig. 2 displays on the console 17 of the designer C a dialog box (not illustrated) that contains the list of all the classes already defined. The designer C can select one of these, which will be the class of the new object. Within an object, values can be given to the attributes. These values will represent its identity and its state.

Furthermore, the global generic class "GenericClass" as it appears in Fig. 3 preferably has an additional attribute consisting in a boolean 0" or "1," which indicates whether the instance of the current generic class represents a class or an object. In the example illustrated, the boolean is "1" to indicate that the designer is creating a class. The boolean is automatically set to the corresponding value in response to the command "New class" or "New class object" that has been activated by the designer C in the window 22 of Fig. 2. In the example chosen, this attribute is not visible to the designer C and is therefore shown in gray.

Also, in the window 21 of Fig. 2, the "Search" menu is very useful to the designer C for performing a powerful search in the base 15 to find classes and objects created in accordance with a list of criteria that he defines. The designer C can, for example, create a new class through inheritance from a class he has found, instead of starting from zero and redoing a job that has already been done.

More generally, the dialog boxes 23-26 are used by the designer, respectively, to generate the global generic class and the generic classes that can compose it. Although the dialog boxes illustrated completely define these classes, not all of the attributes, types, text fields and areas illustrated are necessary. In particular, the descriptions are illustrated for purposes other than the method for automatically generating classes that has just been described.

The description will now refer to the tool 30 for deriving a class or an object, the tool 30 being contained in the module 13 of the command interface 11 of Fig. 1.

Inheritance with the command "New inherited class," instantiation with the command "New class object" and cloning with the command "Clone an object" all require the dynamic creation of a copy of the class (for inheritance or instantiation) or a copy of the object (for cloning).

Fig. 7 also illustrates a tree 27b of a class "ClassD1" obtained by duplicating "Class1." This figure illustrates, in a general way, that the duplication of the instance of "GenericClass" is not enough, since the duplicated class "ClassD1" will point to the same instances "GenericMethod" and "GenericAttribute" as the instance "Class1" of "GenericClass." For example, it is clear from Fig. 7 that "Attribute1" is common to both classes, while the two classes must be distinct and independent. Consequently, in order to derive a class or an object, i.e., in order to create a new class through inheritance, instantiate a class or clone an object, it is necessary not only to duplicate the instance of "GenericClass," but also to duplicate each instance directly or indirectly referenced by the instance of "GenericClass."

Furthermore, software developers use a serialization technique that is applied to many languages, such as Java or C++. Serialization makes it possible to store any object on a data recording medium, a disk for example. If for example a first object references objects, which themselves reference other objects and so on, one need only write the first object onto the disk with the serialization method in order to automatically store in memory the tree of all the objects directly or indirectly referenced by the first object.

Fig. 8 schematically illustrates the principle of the method for deriving a class "Class1" or an object. The tool 30 implements this method. The example illustrated refers to the creation of a class "ClassD2" by inheritance from the class "Class1" as illustrated in Fig. 7, it being understood that the method can be applied in the same way to the instantiation of a class or the cloning of an object. As indicated in Fig. 8, the derivation method consists, in a first step, of serializing the class "Class1" by copying it onto a disk D, for example the hard disk of a machine 2a or 2b of the computer infrastructure 1 of Fig. 1. Thus, the entire tree 27a of this class, as illustrated in Fig. 7, will be duplicated and stored on the disk. The second step consists of reading the duplicated class stored in the disk D, by loading it into the tool 30, i.e., into the RAM in which the program of the tool is loaded. Thus, the duplicated class "ClassD2" has a tree 27c identical to that of the tree 27a of the mother class "Class1" but is independent of the mother

class. In Java language for example, the two steps of the method can be executed for example by the following two instructions:

***FileStream*.writeObject(Class1);**
// for serializing the source tree Class1;
New object or New class = ***FileStream*.readObject();**
// to have a copy of Class1

Fig. 9 illustrates a dialog box 31 for the utilization of the tool 30. This example is similar to that of Fig. 8, which relates to the creation of a class inheriting from another class. Upon execution of the command "New inherited class" from the "Classes" menu of Fig. 3, the dialog box 31 of Fig. 9 appears. This box contains a text area "Name of the new class" and a list of the classes from which the mother class can be chosen. In the example chosen, the class "Class1" is chosen from the list, as indicated in gray in Fig. 9. In the preceding example, the inherited class is named "classD2."

Fig. 10 illustrates an example of a dialog box 32 resulting from the validation of the box 31. The box 32 is similar to the box 23 of Fig. 3. Consequently, only the differences will be indicated below. The data contained in the box 32 is automatically filled in by the software of the tool 30. The name of the class is the one indicated by the designer C in the box 31 of Fig. 9. The box 32 also contains a table indicating the parent or parents of the class, in this case "Class1," which the designer C has selected from the list in the box 31. The other data is similar to that of the box 23, since the tree 27c of the daughter class "ClassD2" is the same as the tree 27a of the mother class "Class1." Using the command buttons in the box 32, it is possible to have only the new class "ClassD2" evolve, independently from the mother class "Class1."

The global generic class "GenericClass" therefore has an additional attribute named "parent," which is a table containing instances of the global generic class, such as Class1. The table makes it possible to know the provenance of the current instance, in the following way:

(a) If the current instance is an object, then:
- if in general "parent[0]" (indicated for example in the table to which the current instance refers) is an object, the current object has been cloned from parent[0],

- if "parent[0]" is a class, the current object is an instance of parent[0],

(b) If the current instance is a class, then "parent[0]" is

- either empty, which means that the class has been created without inheritance with the command "New class,"

- or a class (and not an object), which means that the current class has inherited from "parent[0]" through the command "New inherited class."

(c) With the result, by iteration, that the table "parent" indicates all the ascendants of the current instance.

The method is clearly shown in Figs. 8, 9 and 10. In response to the first instruction indicated in Fig. 8, "Class1" is serialized by being copied onto the disk D. In response to the second instruction, the class thus serialized is again saved in memory, but with modifications of attributes, specifically the class name "ClassD2" and the table "Parent." The two trees 27a and 27c are therefore the same, but they are separate from one another. For example, even though they have the same two instances "Attribute1," in reality these two instances are completely separate from one another in memory and can exist independently from one another. In particular, they can be modified differently from one another at any time. This example also makes it clear that the same is true for the other two commands "New class object" and "Clone an object."

More generally, it is clear from the preceding description that the method for deriving an object class and/or an object having a given name consists of making a copy of the entire tree of the class or the object, saving the copy of the tree and changing said name in order to assign a name to the saved copy. We have seen that the copy is preferably made through a serialization of the tree representing said class or said object by copying the tree into a memory D, and the saving of the copy of the tree consists of copying it again into a memory 30. Furthermore, we have also seen that the serialization, which can be done in various languages, is particularly simple to implement in Java® language.

The function of the interface generator 14 will now be described. Up to this point, we have seen how the designer C can easily create object classes and objects that meet the needs of the user U. The classes and objects thus created are stored in the interface base 15. However, these classes and objects are still incomprehensible and unusable for the user U. Preferably, the

user U is also prevented from accessing them, so that the designer can be assured of their integrity. The function of the generator 14 is to transform the classes and objects thus created into interfaces in the form of documents in which the commands that meet the needs expressed by the user U are understandable to him. In order to distinguish these interfaces from the other interfaces involved in this description, they will be called document interfaces. The language used in these document interfaces can be the current language and/or a specialized language in the user's field of expertise.

The operation of the generator 14 will be explained in the following example, which refers to and elaborates on the preceding example in which the company is a bank branch, the user U is the manager of the branch and the designer C is a computer expert of the bank. It is assumed that the manager U wants to (1) consult the bank's computer infrastructure to query the databases in order to obtain a list of his clients of a given type, students in the example in question, and (2) to print personalized letters of several types, such as information letters, invitation letters, and reminder letters. The computer expert C translates the operation desired by the manager into two questions. In this case, the computer expert creates in response two object classes, "consult_system" and "print," using the two tools 20 and 30. He can create each of these two classes directly by activating the command "New class" in the window 22 of Fig. 2, or indirectly through derivation from a similar class. In the latter case, the designer C can activate the "Search" command in the window 22 to find a similar class. If a similar class exists, he selects it from the list of classes and can derive it, for example by activating the command "New inherited class" of Fig. 2 and by modifying, adding or deleting attributes and/or methods.

The generator 14 implements a process for generating a document interface for the control of a computer system by a user, the command being created from at least one object that includes descriptions. The process consists of extracting at least some of said descriptions from the object and of organizing them so as to translate the meaning of said command into a language understandable to the user and thus create from said interface a document interface. The generator 14 therefore comprises an extraction block 14a and a generation block 14b. The extraction block 14a takes the object selected by the designer from the interface base 15 and extracts the descriptions from it. The generation block 14b organizes the descriptions to create the document interface and stores it in the interface base 15.

The method will be better understood from the preceding example. Among the attributes of the class "consult_system" of the example in question are methods that use specific commands to operate on the data of the system, in this case in the databases of the system 10. From these methods, the designer C creates a method "list_students" in order to have the description "List of all students having...". The engineer assigns to the method "list_students" a return code of the "list" type having the description "list of the names and addresses." He also assigns to this method, using the dialog boxes 25 and 26 illustrated in Figs. 5 and 6, the three parameters defined in Table A below.

name	description	type	default value
"age<"	"an age less than"	Integer	26
"place of study"	", studying in"	Paris, Versailles	Paris
"loan>"	"and having taken out a loan of more than"	Integer	6,000 francs

TABLE A

For the class "print," the engineer C creates a method "print_invitation" in order to have a description such as "Print letters to:" and assigns this method a return code of the "void" type, indicating that the method does not return a value, and two parameters as defined in Table B below.

name	description	type	default value
"addressee"	"Print letters to"	list	list_students of consult_system

"type of letter"	"type of letter:"	birthday invitation; reminder; information on the student loan	information

TABLE B

These tables indicate how to fill in the dialog boxes 25 and 26 of Figs. 5 and 6 so as to constitute the two classes "consult_system" and "print" using the dialog box 23 of Fig. 3. More generally the document interface of an object is created from descriptions corresponding to this object, its attributes, its methods and the parameters of the methods.

To create a document interface, the designer activates the menu "Export to U" in the window 22 of Fig. 2. This command calls the generator 14 for generating a document interface from the selected object. The generator 14 extracts the descriptions of the object and organizes them to create the document interface.

Fig. 11 illustrates the document interface 18a of the method "list_students." It shows that the document interface 18a has been obtained by extracting descriptions from the method and its parameters, as defined in Table A. In particular, the values "26," "Paris," and "6,000 francs" are the default values indicated in Table A. Generally, a document interface 18 comprises text and at least one possible field 19 whose initial content is made of default values and can be modified by the user U.

Fig. 12 illustrates an example of a document interface 18b of the method "print_invitation" of the class "print" defined previously. The examples of Figs. 11 and 12 are enough to allow one skilled in the art to know how to construct a document interface related to an entire object. Generally, when the object contains all the descriptions, they are preferably organized by the generator 14 in the following order:

1. the description of the object (not illustrated), for example "print";

2. the description of each attribute (not illustrated), for example the number of the printer, the print quality, the color of the ink, the printing paper; this description is followed by a

field that corresponds to its value; if the attribute has a limited number of values, this field is a list containing all the possible values and allowing only the selected value to appear;

3. the description of each method (see Tables A and B, Figs. 11 and 12 and the text related to them), this description being attached to and preferably followed by:

3.1 the description of its return value, attached to or followed by a field that represents this value, and

3.2 the description of each of its parameters, attached to or followed by a field (19) representing the value of the parameter;

4. a control means (see Fig. 12) indicating when the method should be executed, which execution can be immediate or deferred, or can occur at a moment determined by a date and a time, or as a result of another defined method.

4.1 In the first case, a command button is labelled "Immediate execution"

4.2 In the second case, a command button includes a label with a value "at" (for example -- print "at" --) followed by a "time" field and a "date" field linked to a calendar of openable days of the year;

4.3 In the third case, a command button includes a label entitled "after the end of execution of" followed by a field that makes it possible to choose a method of any object and that means the method will be executed after the end of the execution of the selected method.

The designer C has the capability to change the presentation of the page, add or remove fields, select the descriptions to be included, and modify the texts of descriptions and types to make the document easier to understand. It follows that the same object can generate several document interfaces adapted to specific needs offered in the object. For example, we have seen above that the object "print" can have a large number of attributes and methods, offering a wide choice of document interfaces as a function of the attributes and methods adapted to needs. It is of course recommended that the designer C create the document interfaces 18 with the help of the user U.

The description will now refer to the command module 12 used by the user U to obtain the document interfaces 18 from the base 15. The module 12 contains a block 40 for generating documents 41 created from document interfaces stored in the base 15. The block 40 is therefore

connected to the base 15 in order to take the document interfaces 18 from it and determine which command documents 41 are adapted to the particular needs of the user U. For example, assuming that the user needs the document "print," an interface that includes the method illustrated in Fig. 12 will be presented to the user, who can modify the values of the fields 19 and select the execution command buttons to create a document 41. The document 41 illustrated in Fig. 12 corresponds to the document interface 18, in which the values of the fields 19 have remained unchanged by the user and the execution button has been deferred to a predetermined time has been activated (in gray in Fig. 12).

Fig. 13 illustrates an example of a dialog box 42 presented to the user U by the user module 12. The menu bar of the box 42 contains, in particular, two main menus, "Documents" and "Preferences." In the "Documents" menu, there is the "Search" command for finding document interfaces, for example from descriptions of the corresponding objects, and the "Open" command for listing document interfaces by the names of their corresponding objects and selecting an object name from them. A selected object is displayed with the description of the object. From this interface, the user creates the desired document. In the "Documents" menu, there are also of course the commands for saving (Save and Save as...) and for closing the document.

The "Preferences" menu contains, in particular, two commands, "Display" and "Display history." The "Display" command displays all of the documents created by the user in the order of execution of the commands he has chosen. This set defines the user's preference. He also has a global view of the methods he has activated from several documents. He can validate it or modify it. For example, clicking twice on a chosen method causes the opening of the corresponding document for possible modifications. The command "Display history" presents all the methods that have been executed by the user, the end of execution status, and the time. The menu also contains the save and close commands.

CLAIMS

1. Method for deriving a class and/or an object having a given name (class1), characterized in that it consists of making a copy (27c) of the entire tree (27a) of the class or the object, of storing (D) the copy of the tree and of changing said name in order to assign a name (class D2) to the stored copy.

2. Method according to claim 1, characterized in that the copy is made through a serialization of the tree representing said class or said object by copying the tree into a memory (D), and the storage of the copy of the tree consists of copying it again into memory (30).

3. Method according to claim 1 or 2, characterized in that the derivation is an inheritance of the class (class1).

4. Method according to claim 1 or 2, characterized in that the derivation is an instantiation of the class (class1).

5. Method according to claim 1 or 2, characterized in that the derivation is a cloning of an object.

6. Method according to any of claims 1 through 5, characterized in that it consists of automatically generating the class or the derived object by means of a tool (30) having at least one dialog box (21).

7. Method according to claim 6, characterized in that it is implemented by a designer (C) who is a computer expert, using a command interface (11) of the computer system (10) used for the control of the computer system by a user (U) who may not be a computer expert.

8. Computer system (10), characterized in that it implements the method defined according to any of claims 1 through 7.

9. System according to claim 8, comprising a computer system (1) and a command interface (11), characterized in that it is implemented in the command interface.

10. System according to claim 9, characterized in that it is implemented in a design module (13) of the command interface (11) by a designer (C) who is a computer expert, using a console (17) used for the control of the computer system by a user (U) who may not be a computer expert.

ABSTRACT

An object class and/or an object having a given name (class1) is derived by making a copy, preferably through serialization, of the entire tree (27a) of the class or the object, by storing the copy of the tree on a disk D and by assigning a name (classD2) to the stored copy.

Fig. 8 to be published.

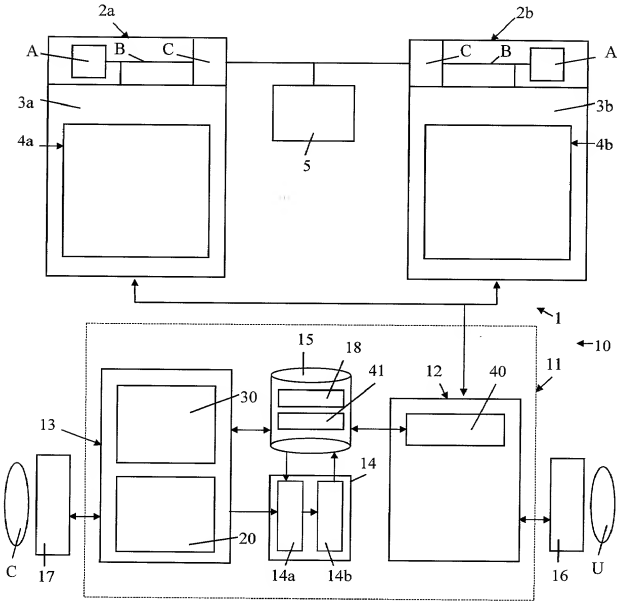

FIG. 1

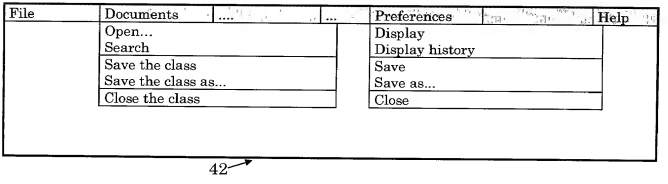

FIG. 13

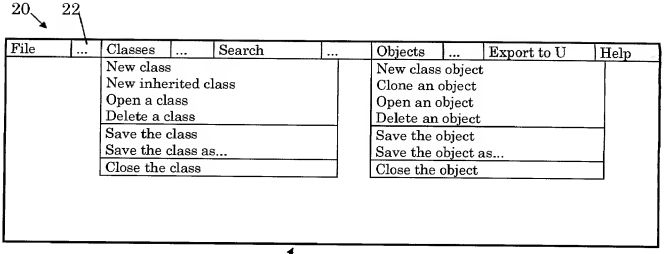
20
22
File
...
Classes
...
Search
...
Objects
...
Export to U
Help
New class
New inherited class
Open a class
Delete a class
Save the class
Save the class as...
Close the class
New class object
Clone an object
Open an object
Delete an object
Save the object
Save the object as...
Close the object
21

FIG. 2

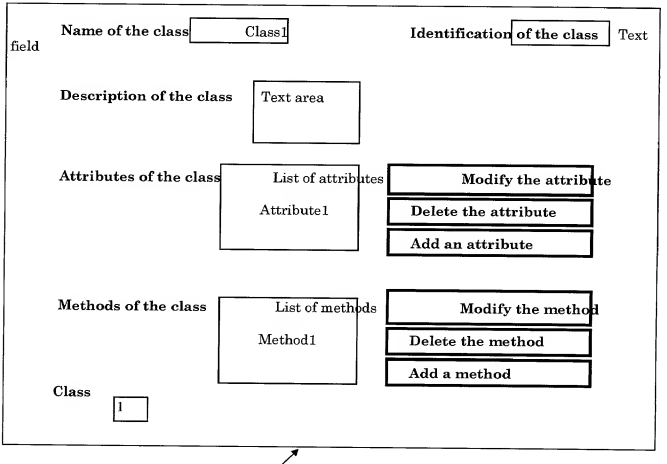
Name of the class
Class1
Identification of the class
Text field
Description of the class
Text area
Attributes of the class
List of attributes
Attribute1
Modify the attribute
Delete the attribute
Add an attribute
Methods of the class
List of methods
Method1
Modify the method
Delete the method
Add a method
Class
1
23
20

FIG. 3

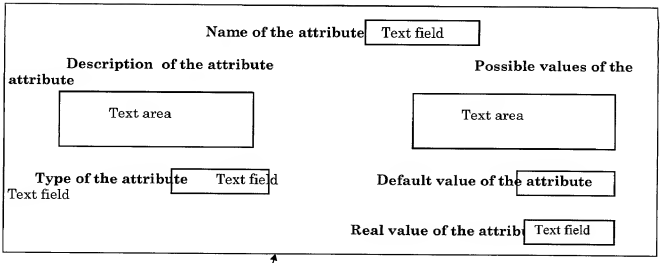

FIG.4

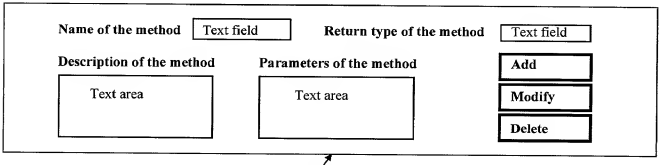

FIG. 5

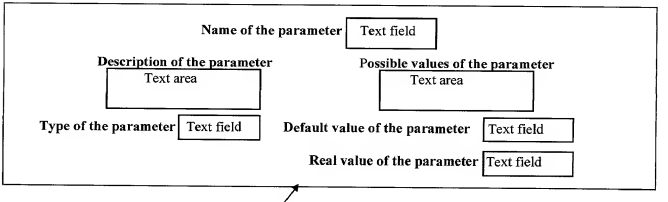

FIG.6

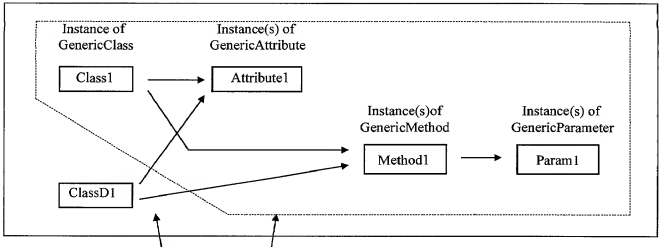

FIG. 7

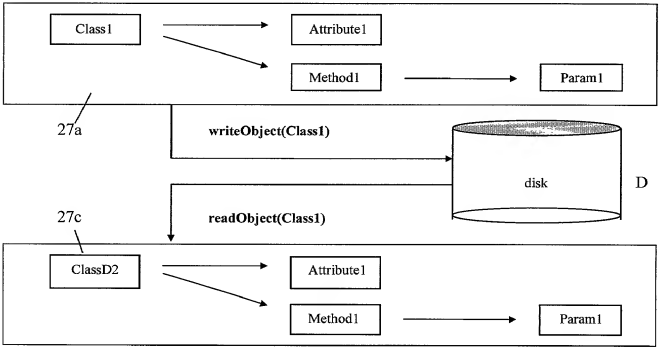

FIG. 8

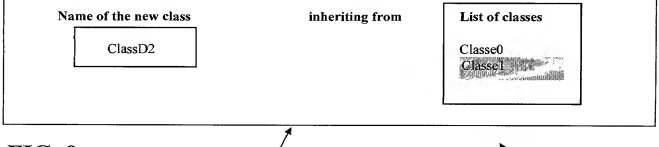

FIG. 9

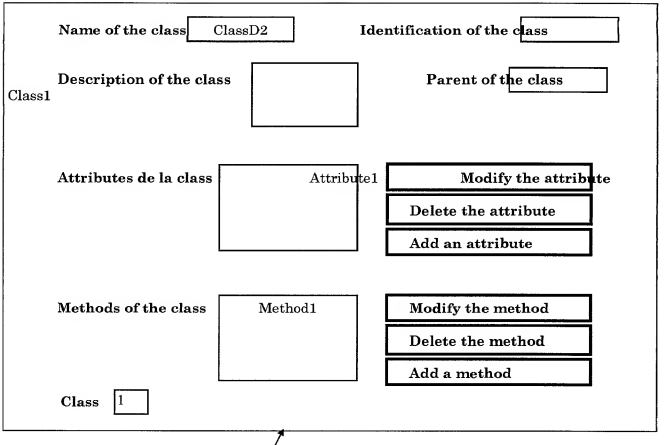

FIG. 10

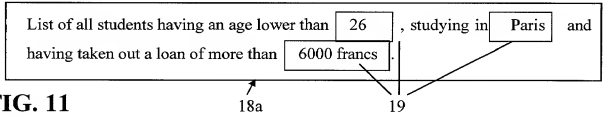

FIG. 11

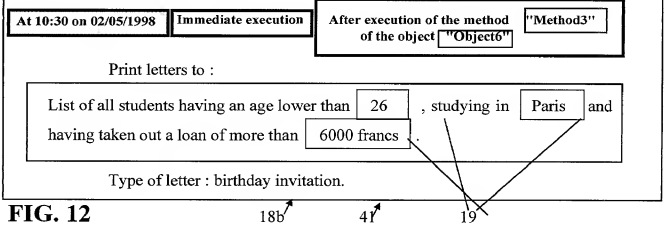

FIG. 12

Declaration and Power of Attorney For Patent Application
Declaration Pour Demandes de Brevets Avec Pouvoirs

French Language Declaration

En tant qu' inventeur nomme ci-après, Je déclare par le présent acte que:

As a below named inventor, I hereby declare that:

Mon nom, mon domicile, mon adresse postale, ma nationalité sont ceux qui figurent ci-après,

My residence, post office address and citizenship are as stated below next to my name,

Je déclare que je crois être l'inventeur original, premier et unique (si un seul nom figure sur le présent acte) ou un des co-inventeurs, originaux et premiers (si plusieurs noms figurent sur le present acte) du sujet revendiqué et pour liquel un brevet est demande sur la base de l'invention intitulée:

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

Dérivation d'une classe d'objets par héritage,
instanciation ou clonage.

dont la description
(cocher la case correspondante)

the specification of which
(check one)

☒ est annexée au présent acte.

☐ a été déposée _____

Numéro de série de la demande _____

et modifiée le _____ (si approprié)

☐ is attached hereto.

☐ was filed on _____ as Application Serial No. _____

and was amended on _____ (if applicable)

Je déclare par le présent acte avoir examiné et compris le contenu de la description identifiée ci-dessus, revendications y compris, et le cas échéant telle que modifiée par l'amendment cité plus haut.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

Je reconnais le devoir de divulguer l'information qui est en rapport avec l'examen de cette demande selon Titre 37 du Code des Reglements Fédéraux §1.56(a).

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

French Language Declaration

Je revendique par le présent acte le bénéfice de priorité étrangère selon Titre 35, du Code des Etats-Unis, §119 de toute demande de brevet ou d'attestation d'inventeur énumérée ci-après, et j'ai identifié également ci-après toute demande étrangère de brevet ou d'attestation d'inventeur ayant une date de dépôt antérieure à celle de la demande pour laquelle la priorité est revendiquée.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior foreign applications

Demande(s) de brevet anterieure(s) dans un autre pays:

(Number) (Numéro)	(Country) (Pays)	(Day/Month/Year Filed) (Jour/Mois/Année de dépôt)	Priority claimed / Droit de priorité revendiqué: Yes / Oui	No / Non
98 13643	FRANCE	30.10.1998	☒	☐
			☐	☐
			☐	☐

Je revendique par le présent acte, le bénéfice selon Titre 35 du Code des Etats-Unis, §120 de toute(s) demande(s) américaines énumérée(s) ci-après et, dans la mesure où le sujet de chacune des revendications de cette demande n'est pas divulgué dans la demande américaine antérieure, de la façon définie par le premier paragraphe de Titre 35 du Code des Etats-Unis, §112, je reconnais le devoir de divulguer l'information pertinente selon Titre 37 du Code des Réglements Fédéraux, §1.56(a), toute information qui se présente entre la date de dépôt de la demande antérieure et la date de dépôt de la demande, soit nationale, soit internationale PCT.

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.) (No. de Demande)	(Filing Date) (Date de Dépôt)	(Etat) (brevetée, pendante, abandonné)	(Status) (patented, pending, abandoned)
PCT/FR 99/02630	28.10.1999	PENDING	

Je déclare par le présent acte que toutes mes déclarations, à ma connaissance, sont vraies et que toutes les déclarations faites à partir de renseignements ou de suppositions, sont tenues pour être vraies; de plus, toutes ces declarations ont été faites en sachant que de fausses déclarations volontaires u autres actes de même nature sont sanctionées par une amende ou un emprisonnement, ou les deux, selon la Section 1001, du Titre 18 de Code des Etats-Unis et que de selles déclarations délibérément fausses peuvent compromettre la validité de la demande ou du brevet délivré.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

French Language Declaration

POUVOIR: En tant qu'inventeur, je désigne l'(les) avocat(s) et/ou l'(les) agent(s) suivant(s) pour poursuivre la procédure de cette demande et traiter toute affaire la concernant supris du Bureau des Brevets et de Marques:

Harold L. Stowell, Reg. 17,233
Edward J. Kondracki, Reg. 20,604
Dennis P. Clarke, Reg. 22,549
William L. Feeney, Reg. 29,918
John C. Kerins, Reg. 32,421

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (*list name and registration number*)

Harold L. Stowell, Reg. 17,233
Edward J. Kondracki, Reg. 20,604
Dennis P. Clarke, Reg. 22,549
William L. Feeney, Reg. 29,918
John C. Kerins, Reg. 32,421

Adresser toure correspondance à:

Edward J. Kondracki, Esq.
KERKAM, STOWELL, KONDRACKI & CLARKE, P.C.
5203 Leesburg Pike, Suite 600
Falls Church, VA 22041

Send Correspondence to:

Edward J. Kondracki, Esq.
KERKAM, STOWELL, KONDRACKI & CLARKE, P.C.
5203 Leesburg Pike, Suite 600
Falls Church, VA 22041

Adresser toute communication téléphonique à: (*Nom*) (*Numéro de téléphone*)

Edward J. Kondracki, Esq.
(703) 998-3302

Direct Telephone Calls to: (*name and telephone number*)

Edward J. Kondracki, Esq.
(703) 998-3302

Nom complet du seul ou premier inventeur **Nachef** Armand	Full name of sole or first inventor
Signature de l'inventeur — Date April 12th, 1999	Inventor's signature — Date
Domicile 6, place Georges Pompidou, 78180 Montigny	Residence
Nationalité Française — le Bretonneux, France	Citizenship
Adresse Postale 6, place Georges Pompidou, 78180 Montigny le Bretonneux, France	Post Office Address
Nom complet du second co-inventeur, le cas echeant **Sitbon** Gérard	Full name of second joint inventor, if any
Signature de l'inventeur — Date April 12, 1999	Second Inventor's signature — Date
Domicile 8, rue Roli, 75014 Paris, France,	Residence
Nationalité Française	Citizenship
Adresse Postale 8, rue Roli, 75014 Paris, France	Post Office Address

(Fournir les mêmes renseignements et la signature de tout co-inventeur supplémentaire.)

(Supply similar information and signature for third and subsequent joint inventors.)